Absolutely! Let’s start **from the very beginning**, assuming you’re a complete beginner, and walk through how to:

✅ Receive a **GitLab Merge Request webhook** using AWS Lambda

✅ Post a **“hi” comment** back to the GitLab Merge Request

We’ll use:

* **AWS Lambda** (to run your code)
* **API Gateway** (to expose your Lambda to GitLab)
* **Python** (for the code)

**✅ WHAT YOU NEED BEFORE STARTING**

1. ✅ An **AWS account** with **free tier** (which you have).
2. ✅ A **GitLab project** (can be a test one).
3. ✅ A **GitLab personal access token** (to allow Lambda to post comments back to GitLab).

**🪜 STEP-BY-STEP GUIDE**

**✅ STEP 1: Get Your GitLab Access Token**

You need this so your code can talk to GitLab and post comments.

1. Go to [GitLab.com](https://gitlab.com)
2. Click your **profile photo** > **Edit Profile** (or “Preferences”)
3. Go to **Access Tokens**
4. Name it something like lambda-token
5. Check the box for **api**
6. Click **Create personal access token**
7. COPY THE TOKEN — you won’t see it again! Save it for later.
8. Error storing to DynamoDB: One or more parameter values were invalid: Type mismatch for key mr\_

*From*

**✅ STEP 2: Open AWS Lambda in AWS Console**

1. Go to <https://console.aws.amazon.com/lambda>
2. Click **Create function**
3. Choose:
   * **Author from scratch**
   * Function name: gitlab-webhook-listener
   * Runtime: **Python 3.12**
4. Click **Create function**

**✅ STEP 3: Add Your GitLab Token to Lambda**

1. After the function is created, go to the **Configuration** tab
2. On the left side, click **Environment variables**
3. Click **Edit**
4. Click **Add environment variable**
   * Key: GITLAB\_TOKEN
   * Value: *paste your GitLab personal access token here*
5. Click **Save**

**✅ STEP 4: Add the Code to Lambda**

1. Go to the **Code** tab
2. Replace the default code with this:

python

CopyEdit

import json  
import os  
import urllib3

http = urllib3.PoolManager()  
GITLAB\_TOKEN = os.environ['GITLAB\_TOKEN']  
GITLAB\_API\_URL = "<https://gitlab.com/api/v4>"

def lambda\_handler(event, context):  
 body = json.loads(event['body'])

print("Webhook received from GitLab")

project\_id = body['project']['id']  
 mr\_iid = body['object\_attributes']['iid']

# Post "hi" comment to GitLab MR  
 url = f"{GITLAB\_API\_URL}/projects/{project\_id}/merge\_requests/{mr\_iid}/notes"  
 headers = {  
 "Content-Type": "application/json",  
 "PRIVATE-TOKEN": GITLAB\_TOKEN  
 }  
 data = json.dumps({"body": "hi"}).encode('utf-8')

response = http.request("POST", url, body=data, headers=headers)

print(f"Posted comment. Status: {response.status}")  
 print(response.data.decode('utf-8'))

return {  
 'statusCode': 200,  
 'body': json.dumps({'message': 'ok'})  
 }

1. Click **Deploy** (top-right)

**✅ STEP 5: Create API Gateway to Expose Your Lambda**

This gives your Lambda function a public URL.

1. In the **Lambda > Configuration** tab, click **Triggers** on the left
2. Click **Add trigger**
3. Select **API Gateway**
   * Choose: Create an API
   * API type: **HTTP API**
   * Security: **Open** (no authentication, fine for now)
4. Click **Add**

🚀 It will give you a **public URL** like this:

arduino

CopyEdit

<https://your-api-id.amazonaws.com/default/gitlab-webhook-listener>

Copy that — you’ll need it in the next step.

**✅ STEP 6: Add Webhook to GitLab**

1. In your GitLab project, go to:
   * **Settings** > **Webhooks**
2. Paste the **API Gateway URL** you just copied
3. Add / at the end (e.g., <https://xyz.execute-api.aws-region.amazonaws.com/default/gitlab-webhook-listener/>)
4. Check the box for **Merge request events**
5. (Optional) Add a secret token for extra security
6. Click **Add webhook**

**✅ STEP 7: Test It! 🎉**

1. In your GitLab project, create a **Merge Request**
2. Go to your **Merge Request page** and scroll to the **comments section**
3. You should see a comment: **hi** — posted by your AWS Lambda!

**✅ YOU DID IT!**

You now have a **serverless GitLab webhook listener** that:

* Receives Merge Request webhooks via AWS Lambda
* Prints a message
* Posts “hi” back as a comment

**✅ Extra Tips (Optional):**

* You can see logs of each Lambda run in **CloudWatch Logs** (useful if something fails).
* Later, you can add secret validation (check GitLab secret token in headers).
* Want to split this into the 4 services later? You can connect this to an SQS queue or another Lambda function.